Estimation of time-series on graphs using Bayesian graph convolutional neural networks

Conference Paper · September 2019

DOI: 10.1117/12.2530046

4 authors, including:

Fatemeh Teimury
McGill University
2 PUBLICATIONS **1** CITATION
SEE PROFILE

Soumyasundar Pal
McGill University
34 PUBLICATIONS **105** CITATIONS
SEE PROFILE

Arezou Amini
McGill University
2 PUBLICATIONS **1** CITATION
SEE PROFILE

Estimation of time-series on graphs using Bayesian graph convolutional neural networks

Fatemeh Teimury, Soumyasundar Pal, Arezou Amini, and Mark Coates

Department of Electrical and Computer Engineering, McGill University, Montreal, Canada

ABSTRACT

There has recently been an explosion of interest in graph neural networks, which extend the application of neural networks to data recorded on graphs. Most of the work has focused on static tasks, where a feature vector is available at each node in a graph, often with an associated label, and the goal is to perform node classification or regression, graph classification, or link prediction. Some work has recently emerged addressing the processing of sequences of data (time-series) on graphs using methods based on neural networks; most strategies involve combining Long Short-Term Memory units (LSTMs) or Gated Recurrent Units (GRUs) with graph convolution. However most of these methods process the observed graph as the ground truth, thereby failing to account for the uncertainty associated with graph structure in the learning task. As a remedy to this issue, in the recently proposed Bayesian graph convolutional neural networks, the provided graph is treated as a noisy observation of a true underlying graph or a realization of a random graph model. We can thus model uncertainty in the identification of relationships between nodes in the graph. We specify a joint posterior probability on the graph and the weights of the neural network and then perform inference through a combination of variational inference and Monte Carlo sampling. In this paper, we extend the Bayesian framework to address a regression task for time-series on graphs.

Keywords: graph convolutional neural network, Bayesian inference, time series estimation, regression

1. INTRODUCTION

There are multiple applications involving data recorded on networks where it is important to infer the attributes and/or labels of some nodes given the attributes and labels of other nodes. In many cases there can be a significant measurement cost associated with collecting data in the network, so it is important to be able to reduce the number of nodes that must be monitored. An example is cellular communication network monitoring, where we wish to track the traffic load at each base station, but we cannot deploy monitors at all base stations, either due to cost of monitor deployment or lack of access. Another example is monitoring task-dependent activity of regions in a brain, where collection of signals can involve invasive implantation of electrodes.

There is a considerable body of work that considers the static task of reconstructing a graph signal.[1–7] These techniques generally exploit an assumption of smoothness of the signal with respect to the graph. In this paper we focus on the reconstruction of time-series. The challenge is to combine spatial information (contemporaneous correlations between the signals at different nodes that are related to the topology of the graph) with temporal information (correlations between subsequent values of the time-series at an individual node).

Several approaches have been adopted to address the reconstruction of time-series on graphs, with some of the most promising being based around kernel approaches.[8,9] Other strategies have included dictionary learning[10] and tracking.[11]For attributed graphs with informative features, graph convolutional neural networks (GCNs)[12,13] have proved effective for reconstruction of static signals, and these have recently been combined with recurrent neural network architectures, including Long Short-Term Memory units (LSTMs) and Gated Recurrent Units (GRUs), to address reconstruction of time-series on graphs and spatio-temporal forecasting.[14–19]

Although these methods have demonstrated promising performance, they rely on an accurate knowledge of the topology of the graph. The GCN-based methods also assume that the correlation structure of the multivariate time-series corresponds to the provided topology. In many cases we are provided with an erroneous or noisy

Corresponding author: Fatemeh Teimury (e-mail: fatemeh.teimury@mail.mcgill.ca)

graph, or there can be a mismatch between the graph topology and the correlation structure. We recently introduced a Bayesian graph convolutional neural network (BGCN) framework that allows us to account for uncertainty in the topology. In essence, we introduce a prior on the graph topology and treat the observed or provided topology as a realization from this model. We can then perform approximate Bayesian inference to jointly estimate the weights of the neural network and the true graph topology.

In this paper, we merge our BGCN approach with GRUs so that we can reconstruct time-series on dynamic graphs. We illustrate the benefits of the Bayesian approach by addressing the task of reconstructing average traffic speeds on road networks.

The remainder of the paper is organized as follows. Section 2 presents the problem statement and Section 3 provides relevant background material. Section 4 presents our methodology. We describe the conducted experiments and report the experimental results in Section 5. Section 6 discusses the results, summarizes our contribution and suggests future avenues of research.

2. PROBLEM STATEMENT

We have access to an observed graph $\mathcal{G}_{obs} = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = N$ nodes, and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ denotes the set of edges. Let $\boldsymbol{x}_{i,t} \in \mathbf{R}^{d_x \times 1}$ be the feature vector at node i at time step $t \in \{1, 2, ..., T\}$ and $\mathbf{X}_t = [\boldsymbol{x}_{1,t}, \boldsymbol{x}_{2,t}, ..., \boldsymbol{x}_{N,t}]^T$ be the collection of them across the nodes. $\boldsymbol{y}_{i,t} \in \mathbf{R}^{d_y \times 1}$ is the target label (time series value) at node i at time step t. The training set at time t is $\mathcal{L}_t \subset \mathcal{V}$ which is constituted of a subset of nodes whose target values at time t are known. These values are denoted by $\mathbf{Y}_{\mathcal{L}_t} = \{\boldsymbol{y}_{i,t} : i \in \mathcal{L}_t\}$. The task is to estimate the unknown target time series values of a test set defined by $\mathcal{V} \setminus \mathcal{L}_t$ for all t.

3. BACKGROUND

3.1 GCN

Graph convolutional neural networks have been applied to a variety of learning tasks including node classification, matrix completion, and learning of node embeddings. In order to provide a concrete description of a GCN, we consider in this section a graph-based static regression problem. In this setting, each node i has a feature vector $\boldsymbol{x}_i \in \mathbf{R}^{d_x \times 1}$ associated with it and its target is denoted by $\boldsymbol{y}_i \in \mathbf{R}^{d_y \times 1}$. The targets are known only for the training set $\mathcal{L} \subset \mathcal{V}$. Based on the observed graph $\mathcal{G}_{obs}$, the feature matrix $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]^T$ and the target values in the training set $\mathbf{Y}_{\mathcal{L}} = \{\boldsymbol{y}_i : i \in \mathcal{L}\}$, we aim to predict the target values at the remaining nodes.

In a GCN, graph convolution operations are performed within a neural network architecture. The propagation rule for the simpler GCN architectures[12,13] can be written as:

$$\mathbf{H}^{(1)} = \sigma(\hat{\mathbf{A}}_{\mathcal{G}} \mathbf{X} \mathbf{W}^{(0)}), \tag{1}$$

$$\mathbf{H}^{(l+1)} = \sigma(\hat{\mathbf{A}}_{\mathcal{G}} \mathbf{H}^{(l)} \mathbf{W}^{(l)}). \tag{2}$$

Here, the normalized adjacency operator $\hat{\mathbf{A}}_{\mathcal{G}}$ is computed based on the observed graph and it controls the mixing of the output features across the neighbourhood of a node at each layer. A pointwise non-linear activation function σ is used between layers. $\mathbf{H}^{(l)}$ are the output features from layer $l-1$. $\mathbf{W}^{(l)}$ represents the weights of the neural network at layer l. We use $\mathbf{W} = \{\mathbf{W}^l\}_{l=1}^L$ to denote the collection of GCN weights across all layers. In an L-layer network, the final output is collected from the last layer $\mathbf{Z} = \mathbf{H}^{(L)}$. The weights of the neural network $\mathbf{W}$ are learned via backpropagation with the objective of minimizing an error metric between the known targets in the training set $\mathbf{Y}_{\mathcal{L}}$ and the network predictions $\mathbf{Z}_{\mathcal{L}} = \{\boldsymbol{z}_i : i \in \mathcal{L}\}$ obtained at the same nodes.

3.2 GRU

A Gated Recurrent Unit (GRU)[20] is a variant of a Recurrent Neural Network (RNN) proposed to allow each recurrent unit to adaptively capture dependencies at different time scales.[21] A GRU processes sequential input

using the following gates:

$$\mathbf{U}_t = \sigma(\mathbf{W}_U \mathbf{I}_t + \mathbf{V}_U \mathbf{S}_{t-1} + \mathbf{B}_U)\,, \tag{3}$$
$$\mathbf{R}_t = \sigma(\mathbf{W}_R \mathbf{I}_t + \mathbf{V}_R \mathbf{S}_{t-1} + \mathbf{B}_R)\,, \tag{4}$$
$$\tilde{\mathbf{S}}_t = tanh(\mathbf{W}_S \mathbf{I}_t + \mathbf{V}_S(\mathbf{R}_t \circ \mathbf{S}_{t-1}) + \mathbf{B}_S)\,, \tag{5}$$
$$\mathbf{S}_t = (1 - \mathbf{U}_t) \circ \mathbf{S}_{t-1} + \mathbf{U}_t \circ \tilde{\mathbf{S}}_t\,. \tag{6}$$

Here $\mathbf{I}_t$ is the input, $\mathbf{S}_{t-1}$ and $\mathbf{S}_t$ are the old and the new states of the GRU. $\mathbf{U}_t$, $\mathbf{R}_t$ and $\tilde{\mathbf{S}}_t$ are the outputs of the update gate, reset gate and pre-output gate respectively. $\mathbf{W}$, $\mathbf{V}$, and $\mathbf{B}$ are learnable parameter matrices and vectors in each of the gates.

3.3 GCN-GRU

Static models are not capable of learning temporal correlation structure from dynamic and spatio-temporal graph datasets. In order to process time-series on graphs, GRUs have been combined within GCN. The resultant GCN-GRU uses a GCN and a GRU sequentially to extract spatial and temporal features, respectively. The GCN-GRU model proposed by Zhao et al.[17,18] feeds the input data $\mathbf{X}_t$ to a GCN block in order to extract spatial features. It then passes the output of the GCN, $\mathbf{Z}_t$, as the input of a GRU block in order to learn temporal features. The combined GCN-GRU equations become:

$$\mathbf{Z}_t = GCN(\mathbf{X}_t, \mathcal{G})\,, \tag{7}$$
$$\mathbf{U}_t = \sigma(\mathbf{W}_U \mathbf{Z}_t + \mathbf{V}_U \mathbf{S}_{t-1} + \mathbf{B}_U)\,, \tag{8}$$
$$\mathbf{R}_t = \sigma(\mathbf{W}_R \mathbf{Z}_t + \mathbf{V}_R \mathbf{S}_{t-1} + \mathbf{B}_R)\,, \tag{9}$$
$$\tilde{\mathbf{S}}_t = tanh(\mathbf{W}_S \mathbf{Z}_t + \mathbf{V}_S(\mathbf{R}_t \circ \mathbf{S}_{t-1}) + \mathbf{B}_S)\,, \tag{10}$$
$$\mathbf{S}_t = (1 - \mathbf{U}_t) \circ \mathbf{S}_{t-1} + \mathbf{U}_t \circ \tilde{\mathbf{S}}_t\,. \tag{11}$$

4. METHODOLOGY

Although our problem statement involves regression for time series on graphs, we first present the Bayesian GCN in a static problem setting and then describe the modifications to handle time series.

4.1 BGCN

In many graph based learning tasks, the observed graph fails to represent the true underlying relationship among the nodes. This might be due to the fact that often the observed graph is derived from noisy data or inaccurate modelling assumptions. In the Bayesian graph convolutional neural network framework,[22] the uncertainty in the graph structure is addressed by viewing the observed graph as a random observation from a parametric family of random graphs. We then perform Bayesian inference of the joint posterior of the graph and the GCN weights. Since the observed graph often does not fit a parametric random graph model, we have recently proposed modifications that involve building the posterior distribution of the graphs using a non-parametric model[23] or a node copying procedure.[24] These formulations also allow the use of node features and training labels in the graph inference, in contrast to the approach presented by Zhang et al.[22] Although these algorithms address the node classification task, they can be easily modified to address a regression task. In a BGCN, the posterior distribution of the predictions $\mathbf{Z}$ are given as:

$$p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}},\mathbf{X},\mathcal{G}_{obs}) = \int p(\mathbf{Z}|\mathbf{W},\mathcal{G},\mathbf{X})p(\mathbf{W}|\mathbf{Y}_{\mathcal{L}},\mathbf{X},\mathcal{G})p(\mathcal{G}|\mathcal{G}_{obs},\boldsymbol{\zeta})p(\boldsymbol{\zeta}|\mathcal{G}_{obs},\mathbf{Y}_{\mathcal{L}},\mathbf{X})\,d\mathbf{W}\,d\mathcal{G}\,. \tag{12}$$

The integral in eq. (12) cannot be computed analytically, hence a Monte Carlo approach is adopted. The inference of the BGCN weights are carried out using Monte Carlo dropout. Different choices for the inference of the posterior distribution of graphs are discussed below.

4.1.1 Node copying graph model

In the node copying graph model, we construct a posterior distribution which allows sampling of a random graph by copying the observed graph and then replacing each node's edges with a high probability by the edges of a "similar" node randomly selected from the observed graph. The node features are unaffected during copying. In order to facilitate sampling of a random graph $\mathcal{G}$, we introduce an auxiliary random vector $\boldsymbol{\zeta} \in \{1, 2, ...N\}^N$, where the i-th entry ζ^i is used to denote the node whose edges are to replace the edges of the i-th node in the observed graph. We assume mutual independence for the entries in $\boldsymbol{\zeta}$. A critical part of the construction of this model is the choice of a similarity metric, which impacts how we sample the ζ^js from their posterior distributions. We adopt the following strategy for the sampling procedure. Using the information provided by $\mathbf{X}$, $\mathbf{Y}_{\mathcal{L}}$ and $\mathcal{G}_{obs}$, we construct a symmetric pairwise distance matrix $D(\mathbf{X}, \mathbf{Y}_{\mathcal{L}}, \mathcal{G}_{obs}) \geq \mathbf{0}$, where $D_{i,j}$ denotes the distance between nodes i and j. Then for each node i, we form a set of prospective nodes to be copied in place of node i by including M nodes based on the smallest distance from node i:

$$\mathcal{C}_i = \{l \mid 1 \leq l \leq N, D_{i,l} = D_{i,(j)} \text{ for } 1 \leq j \leq M\}\,. \tag{13}$$

We define the posterior distribution of $\boldsymbol{\zeta}$ by assigning an equal probability to all of the prospective nodes:

$$\begin{aligned} p(\boldsymbol{\zeta}|\mathcal{G}_{obs}, \mathbf{X}, \mathbf{Y}_{\mathcal{L}}) &= \prod_{i=1}^{N} p(\zeta^i|\mathcal{G}_{obs}, \mathbf{X}, \mathbf{Y}_{\mathcal{L}})\,, \\ p(\zeta^i = m|\mathcal{G}_{obs}, \mathbf{X}, \mathbf{Y}_{\mathcal{L}}) &= \begin{cases} \dfrac{1}{M}, & \text{if } m \in \mathcal{C}_i \\ 0, & \text{otherwise}\,, \end{cases} \end{aligned} \tag{14}$$

for $1 \leq i, m \leq N$. Conditioned on $\boldsymbol{\zeta}$ and the observed graph $\mathcal{G}_{obs}$, the sampling of graph $\mathcal{G}$ is carried out by copying the ζ^i'th node of $\mathcal{G}_{obs}$ in the place of the i'th node of $\mathcal{G}$, independently for all $1 \leq i \leq N$ with a high probability. More formally, the generative model for $\mathcal{G}$ is given as:

$$p(\mathcal{G}|\mathcal{G}_{obs}, \boldsymbol{\zeta}) = \prod_{i=1}^{N} \epsilon^{\mathbb{1}_{\{\mathcal{G}_i = \mathcal{G}_{obs,i}\}}} (1-\epsilon)^{\mathbb{1}_{\{\mathcal{G}_i = \mathcal{G}_{obs,\zeta^i}\}}} \tag{15}$$

where $0 < \epsilon \ll 1$ is a hyperparameter and $\mathbb{1}_{\{\mathcal{G}_i = \mathcal{G}_{obs,q}\}}$ denotes the indicator function of copying the edges of the q-th node of $\mathcal{G}_{obs}$ to replace the edges of the i-th node of $\mathcal{G}$.

4.1.2 Non-parametric graph model

In the non-parametric graph model,[23] the prior distribution for a graph $\mathcal{G}$ is defined as:

$$p(\mathcal{G}) \propto \begin{cases} \exp\left(\alpha \mathbf{1}^T \log(A_{\mathcal{G}}\mathbf{1}) - \beta \|A_{\mathcal{G}}\|_F^2\right), & \text{if } A_{\mathcal{G}} \geq \mathbf{0}, A_{\mathcal{G}} = A_{\mathcal{G}}^T \\ 0, & \text{otherwise}\,, \end{cases} \tag{16}$$

where the symmetric adjacency matrix with non-negative entries of the random undirected graph $\mathcal{G}$ is denoted by $A_{\mathcal{G}}$. The first term in the log prior prevents any isolated node in $\mathcal{G}$ and the second encourages low weights for the edges. The scale and sparsity of $A_{\mathcal{G}}$ can be controlled using the hyperparameters α and β. As in Section 4.1.1, we construct a symmetric pairwise distance matrix $D(\mathbf{X}, \mathbf{Y}_{\mathcal{L}}, \mathcal{G}_{obs}) \geq \mathbf{0}$ using the information provided by $\mathbf{X}$, $\mathbf{Y}_{\mathcal{L}}$ and $\mathcal{G}_{obs}$. Based on this, we define the joint likelihood of $\mathbf{X}$, $\mathbf{Y}_{\mathcal{L}}$ and $\mathcal{G}_{obs}$ to be:

$$p(\mathbf{X}, \mathbf{Y}_{\mathcal{L}}, \mathcal{G}_{obs}|\mathcal{G}) \propto \exp\left(-\|A_{\mathcal{G}} \circ D(\mathbf{X}, \mathbf{Y}_{\mathcal{L}}, \mathcal{G}_{obs})\|_{1,1}\right), \tag{17}$$

where the symbol $\circ$ denotes the Hadamard product and $\|\cdot\|_{1,1}$ denotes the elementwise l_1 norm. This likelihood encourages high values in $A_{\mathcal{G}}$ in those positions where the corresponding node pairs have low distance between them.

4.2 BGCN-GRU

The main idea behind the proposed BGCN-GRU is to replace the GCN block in the GCN-GRU architecture by a BGCN block. In order to train the BGCN, we need to perform the inference of the posterior of the graph $\mathcal{G}$. In this setting, we compute the pairwise distance matrix D, based on the predictions of GCN-GRU algorithm:

$$D_{i,j}(\mathbf{X}, \mathbf{Y}_{\mathcal{L}}, \mathcal{G}_{obs}) = \frac{1}{T}\sum_{t=1}^{T} \|s_{i,t} - s_{j,t}\|_2^2 \,, \tag{18}$$

where $s_{i,t}$ is the output of the GCN-GRU node i, at time stamp t. The architecture of the constructed BGCN-GRU is depicted in Figure 1.

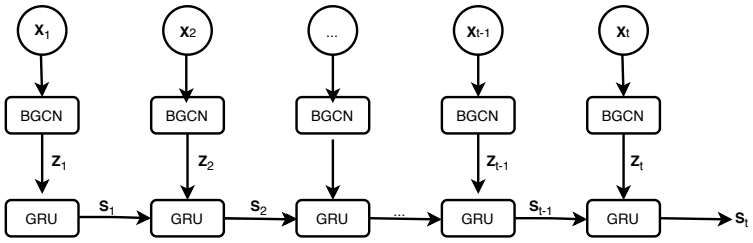

Figure 1: Architecture of the proposed BGCN-GRU.

5. EXPERIMENTAL RESULTS

5.1 Dataset

We conducted experiments on the PEMS-BAY dataset[14] which is a traffic dataset that was collected by the California Transportation Agencies' (CalTrans) Performance Measurement System (PeMS). 325 sensors are selected from the Bay Area. We analyze 6 months of data ranging from Jan. 1st 2017 to May 31st 2017. The graph has an adjacency matrix of size 325×325, which represents the road connectivity. Each timestep represents a 5 minute interval, and the total number of timesteps is thus $T = 52,116$. Since the dataset does not possess any node features, we implement the GCN block in a featureless manner ($\mathbf{X}_t = I$). In our work, in order to provide a more focused comparison and to explore the capacity of the BGCN-GRU to learn from a smaller amount of data, we examine performance for the first 10, 100, and 1000 timestamps.

We run all the experiments for 5 trials and report the average of MAE, RMSE, and MAPE values over all trials. We use 80% of our data as the training set (randomly selected nodes) and 20% as the test set. The results are presented in Table 1.

5.2 Baseline methods

We compare the performance of our proposed methods with the GCN[13] and the GCN-GRU[17,18] algorithms. For the GCN algorithm, we train a separate GCN for each timestep by minimizing the MAE cost function for that particular timestep. The GCN method thus disregards any temporal correlations in the data. When training the GCN-GRU algorithm, we minimize the average MAE over the training set across all of the timesteps. In both cases, the architecture of the GCN consists of 2 layers with a hidden layer of 10 units, and the output is a scalar embedding for each one of the nodes. For all the experiments in Table 1, we set the learning rate to be 0.005 and dropout probability to be 0.2, and we train all of the algorithms for 1000 epochs.

5.3 Evaluation metrics

For all the nodes, let $\mathbf{Y} = \{\mathbf{Y}_t\}_{t=1}^{T}$ and $\mathbf{S} = \{\mathbf{S}_t\}_{t=1}^{T}$ be the prediction and the ground truth for all timestamps respectively. To evaluate the algorithms, we use the following metrics:

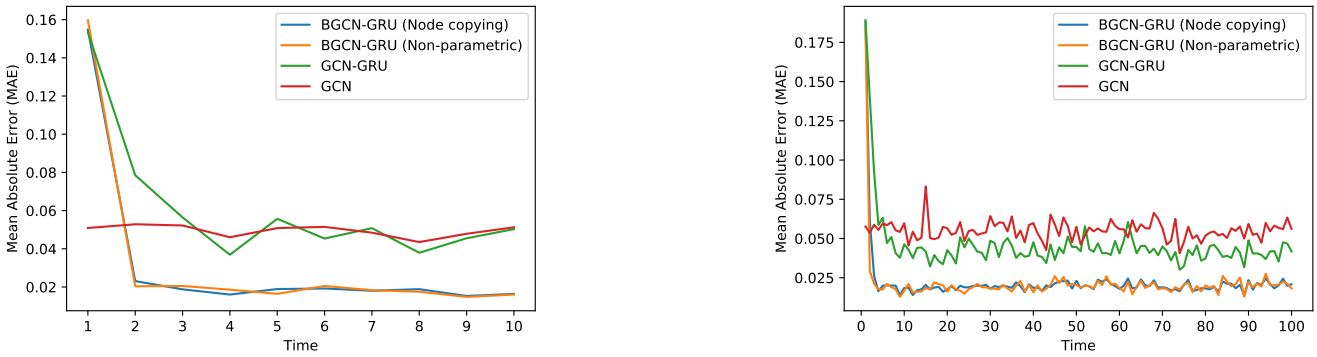

Figure 2: Mean Absolute Error vs. Time (left: 10 timestep scenario; right: 100 timestep scenario)

Root Mean Square Error (RMSE)

$$RMSE(\mathbf{Y}, \mathbf{S}) = \frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{1}{|\mathcal{V} \setminus \mathcal{L}_t|}\sum_{i \in \mathcal{V} \setminus \mathcal{L}_t}(y_{i,t} - s_{i,t})^2}$$

Mean Absolute Percentage Error (MAPE)

$$MAPE(\mathbf{Y}, \mathbf{S}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{V} \setminus \mathcal{L}_t|}\sum_{i \in \mathcal{V} \setminus \mathcal{L}_t}\left|\frac{y_{i,t} - s_{i,t}}{y_{i,t}}\right|$$

Mean Absolute Error (MAE)

$$MAE(\mathbf{Y}, \mathbf{S}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{|\mathcal{V} \setminus \mathcal{L}_t|}\sum_{i \in \mathcal{V} \setminus \mathcal{L}_t}|y_{i,t} - s_{i,t}|$$

Algorithm	**10 timesteps**			**100 timesteps**			**1000 timesteps**		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
GCN	0.0509	0.0685	0.0548	0.0654	0.0877	0.0728	0.0812	0.0996	0.1078
GCN-GRU	0.0613	0.0896	0.0657	0.0454	0.0689	0.0503	0.0623	0.0906	0.1100
BGCN-GRU (Node copying)	0.0326	0.0385	0.0349	0.0184	0.0285	0.0205	0.0544	0.0786	0.1020
BGCN-GRU (Non-parametric)	0.0305	0.0374	0.0325	0.0184	0.0283	0.0205	0.0506	0.0754	0.0981

Table 1: Average MAE, RMSE, and MAPE for prediction on PEMS-BAY dataset

From Table 1, we observe that the proposed BGCN-GRU algorithms have a significantly reduced estimation error for the 10 and 100 timestep scenarios. In the longer sequence of 1000 timesteps, there are multiple periods where the time-series exhibit dramatic changes; all of the estimation algorithms perform poorly during these periods, so the relative performances are more comparable. Figure 2 depicts how the mean absolute error varies with respect to time for the 10-timestep and 100-timestep scenarios. Both the GCN-GRU and BGCN-GRU algorithms have a high error for the first timestep but thereafter exhibit stable estimation performance for the remaining timesteps.

6. CONCLUSION

We have propose a novel methodology for the task of estimating time series on the nodes of a graph using a Bayesian graph neural network approach. The proposed techniques combine Bayesian GCNs with GRUs. The resulting algorithms accounts for the uncertainty in the graph structure in a principled way and outperforms the non-Bayesian baseline algorithms significantly for moderate sequence lengths. Future research directions include conducting thorough experiments on multiple datasets, handling time varying graphs in the Bayesian framework and extending the methodology to other graph-based sequential learning tasks such as forecasting and graph process inference.

REFERENCES

[1] Kolaczyk, E. D., "Analysis of network data: methods and models," (2009).

[2] Kondor, R. I. and Lafferty, J., "Diffusion kernels on graphs and other discrete structures," in [*Proc. Int. Conf. Machine Learning*], 315–322 (2002).

[3] Smola, A. J. and Kondor, R., "Kernels and regularization on graphs," *Learning Theory and Kernel Machines* , 144–158 (2003).

[4] Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P., "The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains," *arXiv preprint, arXiv:1211.0053* (2012).

[5] Belkin, M., Niyogi, P., and Sindhwani, V., "Manifold regularization: a geometric framework for learning from labeled and unlabeled examples," *J. Machine Learning Research* **7**(Nov), 2399–2434 (2006).

[6] Cortes, C. and Mohri, M., "On transductive regression," in [*Proc. Adv. Neural Info. Process. Sys.*], 305–312 (2007).

[7] Wasserman, L. and Lafferty, J. D., "Statistical analysis of semi-supervised regression," in [*Proc. Adv. Neural Info. Process. Sys.*], 801–808 (2008).

[8] Romero, D., Ma, M., and Giannakis, G. B., "Kernel-based reconstruction of graph signals," *IEEE Trans. Signal Process.* **65**(3), 764–778 (2016).

[9] Romero, D., Ioannidis, V. N., and Giannakis, G. B., "Kernel-based reconstruction of space-time functions on dynamic graphs," *IEEE J. Sel. Topics Signal Process.* **11**(6), 856–869 (2017).

[10] Forero, P. A., Rajawat, K., and Giannakis, G. B., "Prediction of partially observed dynamical processes over networks via dictionary learning," *IEEE Trans. Signal Process.* **62**(13), 3305–3320 (2014).

[11] Wang, X., Wang, M., and Gu, Y., "A distributed tracking algorithm for reconstruction of graph signals," *IEEE J. Sel. Topics Signal Process.* **9**(4), 728–740 (2015).

[12] Defferrard, M., Bresson, X., and Vandergheynst, P., "Convolutional neural networks on graphs with fast localized spectral filtering," in [*Proc. Adv. Neural Inf. Process. Sys.*], (2016).

[13] Kipf, T. and Welling, M., "Semi-supervised classification with graph convolutional networks," in [*Proc. Int. Conf. Learning Representations*], (2017).

[14] Li, Y., Yu, R., Shahabi, C., and Liu, Y., "Diffusion convolutional recurrent neural network: data-driven traffic forecasting," in [*Proc. Int. Conf. Learning Representations*], (2018).

[15] Zhang, J., Shi, X., Xie, J., Ma, H., King, I., and Yeung, D.-Y., "GaAN: Gated attention networks for learning on large and spatiotemporal graphs," in [*Proc. Int. Conf. Uncertainty in Artificial Intelligence*], (2018).

[16] Yu, B., Yin, H., and Zhu, Z., "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in [*Proc. Int. Joint Conf. Artificial Intell.*], 3634–3640 (2018).

[17] Zhao, L., Song, Y., Deng, M., and Li, H., "Temporal graph convolutional network for urban traffic flow prediction method," *arXiv preprint, arXiv:1811.05320* (2018).

[18] Zhao, X., Chen, F., and Cho, J. H., "Deep learning for predicting dynamic uncertain opinions in network data," in [*Proc. IEEE Int. Conf. Big Data*], 1150–1155 (2018).

[19] Ruiz, L., Gama, F., and Ribeiro, A., "Gated graph convolutional recurrent neural networks," *arXiv preprint, arXiv:1903.01888* (2019).

[20] Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y., "On the properties of neural machine translation: Encoder-decoder approaches," *CoRR* **abs/1409.1259** (2014).

[21] Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y., "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR* **abs/1412.3555** (2014).

[22] Zhang, Y., Pal, S., Coates, M., and Üstebay, D., "Bayesian graph convolutional neural networks for semi-supervised classification," in [*Proc. AAAI Conf. Artificial Intelligence*], (2019).

[23] Pal, S., Regol, F., and Coates, M., "Bayesian graph convolutional neural networks using non-parametric graph learning," in [*Representation Learning on Graphs and Manifolds Workshop, Int. Conf. Learning Representations*], (2019).

[24] Pal, S., Regol, F., and Coates, M., "Bayesian graph convolutional neural networks using node copying," in [*Learning and Reasoning with Graph-Structured Representations Workshop, Int. Conf. Machine Learning*], (2019).